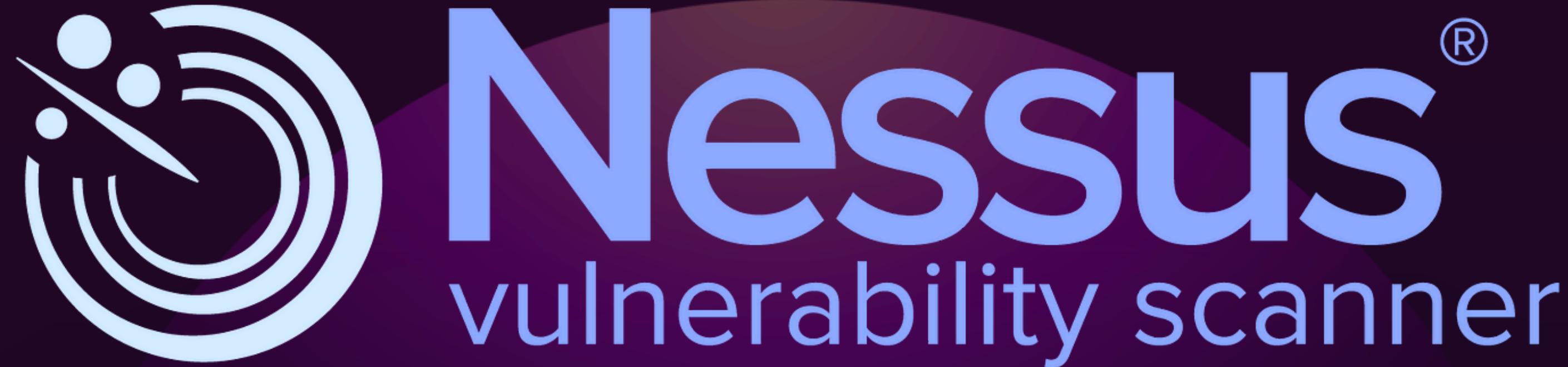


SS-L5

RISOLUZIONE DI VULNERABILITÀ
CRITICHE TROVATE SU
METASPLOITABLE ATTRAVERSO UNA
SCANSIONE CON NESSUS

CHE COS'E NESSUS?



Nessus è uno dei più noti e potenti scanner di vulnerabilità attualmente disponibili sul mercato.

Uno scanner di vulnerabilità è uno strumento progettato per identificare e valutare le vulnerabilità di sicurezza presenti in sistemi informatici, reti e applicazioni. Questi strumenti eseguono scansioni automatizzate per individuare possibili punti deboli che potrebbero essere sfruttati da malintenzionati per compromettere la sicurezza dei sistemi.

Caratteristiche Principali

Nessus esegue scansioni automatiche delle reti, dei sistemi e delle applicazioni alla ricerca di vulnerabilità, configurazioni errate e potenziali minacce alla sicurezza.

È alimentato da un ampio database di vulnerabilità che viene costantemente aggiornato.

Questo contiene informazioni dettagliate su migliaia di vulnerabilità note, consentendo a Nessus di identificare i rischi potenziali in base alle configurazioni e alle versioni del software.

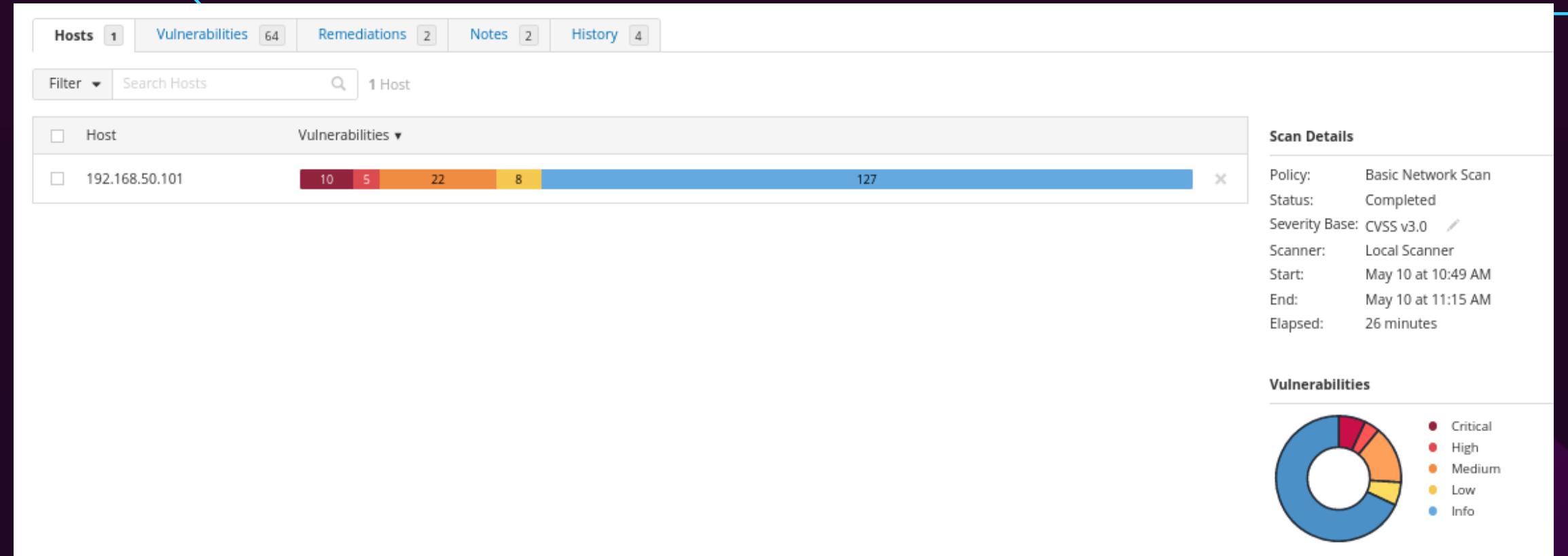
Può essere integrato con altri strumenti di sicurezza e sistemi di gestione per automatizzare processi di mitigazione e risposta agli incidenti.

Supporta una vasta gamma di sistemi operativi, dispositivi di rete e applicazioni, consentendo agli utenti di analizzare l'intera infrastruttura IT per individuare eventuali punti deboli.

Classifica le vulnerabilità individuate in base alla gravità, consentendo agli amministratori di concentrarsi sulle minacce più critiche e urgenti per la sicurezza.

Consente agli utenti di personalizzare le scansioni in base alle proprie esigenze, definendo quali programmi utilizzare, escludendo determinati hosts o reti, possono essere configurati criteri di vulnerabilità.

Fornisce report dettagliati sulle vulnerabilità individuate, comprese descrizioni, raccomandazioni di mitigazione e rischi associati.



Una volta completato lo scan sulla macchina target Nessus ci mostrerà , con un grafico molto intuitivo, le vulnerabilità che ha trovato. Il tool fa uso del suo database che contiene una grande quantità di informazioni di vulnerabilità note; così facendo riesce a identificare e assegnare un punteggio di rischio alle vulnerabilità trovate.

Nessus ha identificato:

- 10 vulnerabilità di livello CRITICAL
- 5 vulnerabilità di livello HIGH
- 22 vulnerabilità di livello MEDIUM
- 8 vulnerabilità di livello LOW
- 127 INFO

Sev	CVSS	VPR	Name	Family
CRITICAL	10.0 *	5.9	NFS Exported Share Information Disclosure	RPC
CRITICAL	10.0		Unix Operating System Unsupported Version Detection	General
CRITICAL	10.0 *		VNC Server 'password' Password	Gain a shell remotely
CRITICAL	9.8	9.0	Apache Tomcat AJP Connector Request Injection (Ghostcat)	Web Servers
CRITICAL	9.8		SSL Version 2 and 3 Protocol Detection	Service detection
CRITICAL	9.8		Bind Shell Backdoor Detection	Backdoors
CRITICAL	SSL (Multiple Issues)	Gain a shell remotely
HIGH	7.5	5.9	Samba Badlock Vulnerability	General
HIGH	7.5		NFS Shares World Readable	RPC
MIXED	SSL (Multiple Issues)	General
MIXED	ISC Bind (Multiple Issues)	DNS

Nella scheda “Vulnerabilities” Nessus ci mostrerà una lista con tutte le vulnerabilità presenti nella macchina target indicandone la pericolosità (Critical, High etc.). Cliccando su una vulnerabilità specifica Nessus aprirà una pagina dove fornirà il nome della vulnerabilità, la descrizione, suggerimenti per la risoluzione, e materiale informativo riguardo la vulnerabilità.

VNC Server

CRITICAL VNC Server 'password' Password

Description
The VNC server running on the remote host is secured with a weak password. Nessus was able to login using VNC authentication and a password of 'password'. A remote, unauthenticated attacker could exploit this to take control of the system.

Solution
Secure the VNC service with a strong password.

Output

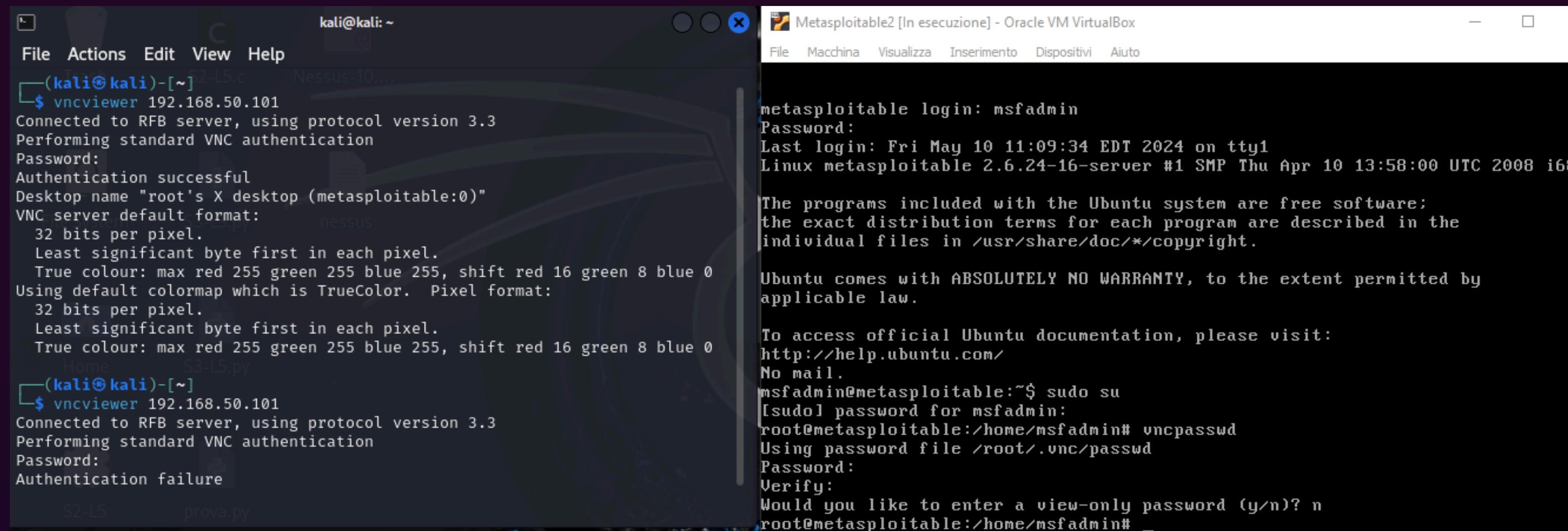
```
Nessus logged in using a password of "password".  
To see debug logs, please visit individual host  
Port ▲ Hosts  
5900 / tcp / vnc 192.168.50.101
```

Tra le vulnerabilità critiche rilevate da Nessus troviamo che la password del servizio VNC Server, che permette l'accesso da remoto, quindi una backdoor, ha una password decisamente troppo semplice, in questo caso "password".

Come dimostrato in figura inserendo la password abbiamo accesso alla macchina, risultando **root** all'interno di Metasploitable, potendo effettivamente fare tutto ciò che vogliamo alla macchina, potenzialmente causando danni incalcolabili.

```
(kali㉿kali)-[~] 52-L5.c$ vncviewer 192.168.50.101  
Connected to RFB server, using protocol version 3.3  
Performing standard VNC authentication  
Password:  
Authentication successful  
Desktop name "root's X desktop (metasploitable:0)"  
VNC server default format:  
    32 bits per pixel.  
    Least significant byte first in each pixel.  
    True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0  
Using default colormap which is TrueColor. Pixel format:  
    32 bits per pixel.  
    Least significant byte first in each pixel.  
    True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
```

VNC Server



The image shows two terminal windows side-by-side. The left window is running on a Kali Linux host (kali@kali) and connects to a VNC server at 192.168.50.101 using vncviewer. It shows the standard VNC authentication process, where a password is entered and the connection is successful, displaying the root's X desktop of the Metasploitable2 target. The right window is running on the Metasploitable2 target (msfadmin@metasploitable) and shows the user attempting to log in with the password 'msfadmin'. The login fails, and the user then runs the vncpasswd command to change the password. The password is successfully changed, and the user attempts to log in again, which fails, demonstrating that the vulnerability has been resolved.

```
kali@kali:~$ vncviewer 192.168.50.101
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
(kali㉿kali)-[~]
$ vncviewer 192.168.50.101
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication failure

Metasploitable login: msfadmin
Password:
Last login: Fri May 10 11:09:34 EDT 2024 on ttym1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/**/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ sudo su
[sudo] password for msfadmin:
root@metasploitable:/home/msfadmin# vncpasswd
Using password file /root/.vnc/passwd
Password:
Verify:
Would you like to enter a view-only password (y/n)? n
root@metasploitable:/home/msfadmin#
```

Per risolvere il problema è necessario cambiare la password che regola l'accesso a questo servizio, è possibile farlo attraverso il comando **vncpasswd** impostando una qualsiasi altra password (possibilmente più difficile da indovinare), e ritentando ad accedere alla backdoor, possiamo notare che l'autenticazione fallisce, dimostrando che la vulnerabilità è stata risolta.

Bind Shell Backdoor

CRITICAL Bind Shell Backdoor Detection

Description
A shell is listening on the remote port without any authentication being required. An attacker may use it by connecting to the remote port and sending commands directly.

Solution
Verify if the remote host has been compromised, and reinstall the system if necessary.

Output

```
Nessus was able to execute the command "id" using the following request :  
  
This produced the following truncated output (limited to 10 lines) :  
----- snip -----  
root@metasploitable:/# uid=0(root) gid=0(root) groups=0(root)  
root@metasploitable:/#  
  
----- snip -----  
  
To see debug logs, please visit individual host  
Port ▲ Hosts  
1524/tcp / wild_shell 192.168.50.101
```

Effettuando una semplice scansione con **nmap** possiamo vedere qual è la porta a cui è connessa e vedere che si tratta effettivamente di una bindshell che permetterebbe un accesso **root**.

Nessus ci informa che è presente un'altra backdoor, una shell legata (bind) a una porta, che può quindi facilmente essere sfruttata ancora una volta per ottenere accesso alla macchina, con strumenti come **netcat** o **telnet**.

```
(kali㉿kali)-[~]$ nmap -sV 192.168.50.101  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-05-10 17:13 CEST  
Nmap scan report for 192.168.50.101  
Host is up (0.0013s latency).  
Not shown: 978 closed tcp ports (conn-refused)  
PORT      STATE SERVICE      VERSION  
21/tcp    open  ftp          vsftpd 2.3.4  
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)  
23/tcp    open  telnet        Linux telnetd  
25/tcp    open  smtp         Postfix smptd  
53/tcp    open  domain       ISC BIND 9.4.2  
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)  
111/tcp   open  rpcbind     2 (RPC #100000)  
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)  
513/tcp   open  login?      Netkit rshd  
514/tcp   open  shell        GNU Classpath grmiregistry  
1099/tcp  open  java-rmi    Metasploitable root shell  
1524/tcp  open  bindshell   Metasploitable root shell  
2049/tcp  open  nfs         2-4 (RPC #100003)  
2121/tcp  open  ftp         ProFTPD 1.3.1
```

Bind Shell Backdoor

```
msfadmin@metasploitable:~$ sudo netstat -tulpn | grep :1524
[sudo] password for msfadmin:
tcp        0      0 0.0.0.0:1524          0.0.0.0:*
LISTEN
4572/xinetd
msfadmin@metasploitable:~$ sudo kill 4572
```

Con l'operatore pipe | andiamo a utilizzare l'output del comando precedente come input per il successivo.

Con grep andiamo a cercare la presenza di “:1524” nell'input, in modo da mostrare solo ciò che ci interessa.

Attraverso un semplice comando netstat andiamo a ottenere informazioni sulla rete, nel nostro caso connessioni attive sui protocolli TCP (t), UDP (u), i sockets in ascolto (l), il PID dei processi (p) e il nome, e l'indirizzo numerico (n). Sarebbero state sufficienti meno opzioni, ma è sempre meglio avere più informazioni.

Infine andiamo a terminare il processo attraverso il suo PID, eliminando la backdoor. Questa è una soluzione temporanea ma efficace.

NFS Exported Shares

CRITICAL NFS Exported Share Information Disclosure

Description
At least one of the NFS shares exported by the remote server could be mounted by the scanning host. An attacker may be able to leverage this to read (and possibly write) files on remote host.

Solution
Configure NFS on the remote host so that only authorized hosts can mount its remote shares.

Output

```
The following NFS shares could be mounted :  
+ /  
+ Contents of / :  
- .  
- ..  
- bin  
- boot  
- cdrom  
more...  
To see debug logs, please visit individual host
```

Port ▲	Hosts
2049 / udp / rpc-nfs	192.168.50.101

Nessus ci informa che c'è la possibilità che un attaccante possa ottenere il contenuto di alcune directories, in questo caso tutte perché è a livello di root.

NFS Exported Shares

```
# Example for NFSv2 and NFSv3:  
# /srv/homes      hostname1(rw,sync) hostname2(ro,sync)  
#  
# Example for NFSv4:  
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt)  
# /srv/nfs4/homes  gss/krb5i(rw,sync)  
  
#  
192.168.50.101(ro,sync,no_root_squash,no_subtree_check)  
  
[ Wrote 12 lines ]  
  
msfadmin@metasploitable:/mnt$ sudo /etc/init.d/nfs-kernel-server restart  
* Stopping NFS kernel daemon [ OK ]  
* Unexporting directories for NFS kernel daemon... [ OK ]  
* Exporting directories for NFS kernel daemon... [ OK ]  
* Starting NFS kernel daemon [ OK ]  
msfadmin@metasploitable:/mnt$
```

```
(kali㉿kali)-[~]  
└─$ sudo mount -t nfs 192.168.50.101:/lib /mnt/local_share  
[sudo] password for kali:  
mount.nfs: access denied by server while mounting (null)
```

Andando a cambiare chi può esportare sostituendo "*" (chiunque) con l'indirizzo IP della/e macchine possono avere l'accesso (uno dei tanti modi per revocare il potenziale accesso a tutti), non rischiamo più che l'attaccante possa montare contenuti sulla sua macchina.
Restartiamo il server tentando di nuovo a ottenere accesso alla macchina possiamo vedere come il processo non vada a buon fine.

Apache Tomcat AJP Connector Request Injection

(Ghostcat)

CRITICAL Apache Tomcat AJP Connector Request Injection (Ghostcat)



Plugin Details

Description

A file read/inclusion vulnerability was found in AJP connector. A remote, unauthenticated attacker could exploit this vulnerability to read web application files from a vulnerable server. In instances where the vulnerable server allows file uploads, an attacker could upload malicious JavaServer Pages (JSP) code within a variety of file types and gain remote code execution (RCE).

Severity: Critical

ID: 134862

Version: 1.44

Type: remote

Family: Web Servers

Published: March 24, 2020

Modified: March 19, 2024

Solution

Update the AJP configuration to require authorization and/or upgrade the Tomcat server to 7.0.100, 8.5.51, 9.0.31 or later.

Apache Tomcat AJP Connector Request Injection

(Ghostcat)

```
noCompressionUserAgents="ozilla, traviata"
compressableMimeType="text/html, text/xml"
-->

<!-- Define a SSL HTTP/1.1 Connector on port 8443 -->
<!--
<Connector port="8443" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" />
-->

<!-- Define an AJP 1.3 Connector on port 8009 -->
<Connector port="8009"
secretRequired="true" secret="Epicode" allow="IP_HOST" />

<!-- Define a Proxied HTTP/1.1 Connector on port 8082 -->
<!-- See proxy documentation for more information about using this. -->
<!--
```

Aggiorna la configurazione AJP:

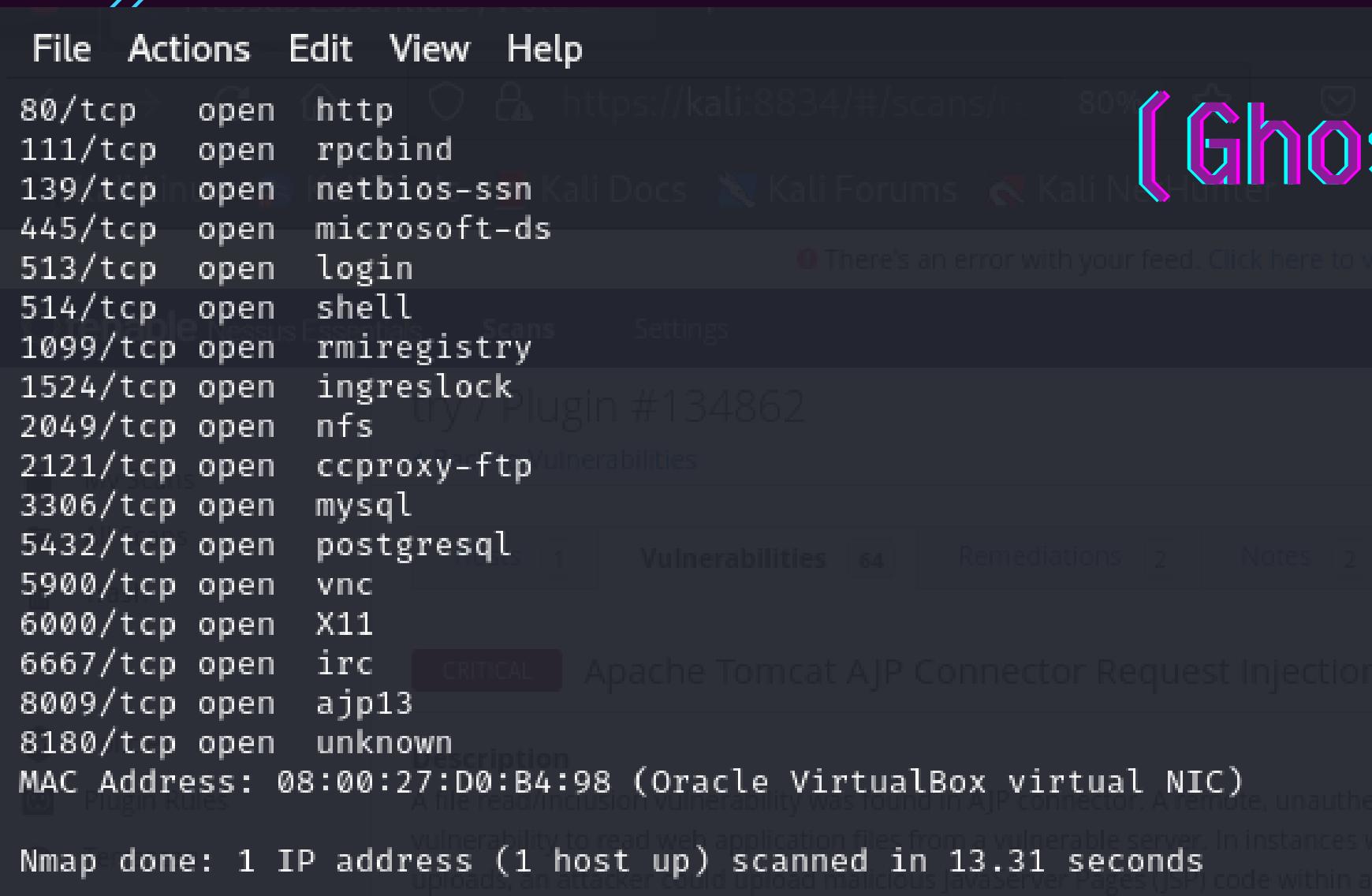
All'interno della sezione di configurazione, aggiungiamo secretRequired, secret, allow.

secretRequire: ci servirà per richiedere l'autenticazione per il connettore AJP

secret: funge da chiave di autenticazione tra Apache e Tomcat. Solo le richieste AJP che includono il parametro secret corretto saranno accettate e processate da Tomcat

allow: specifica un elenco di indirizzi IP o reti consentite a connettersi al connettore AJP. Solo gli indirizzi IP o le reti elencate saranno autorizzati a inviare richieste AJP al server Tomcat. Gli altri indirizzi IP saranno negati.

Apache Tomcat AJP Connector Request Injection



Port	Type	Service
80/tcp	open	http
111/tcp	open	rpcbind
139/tcp	open	netbios-ssn
445/tcp	open	microsoft-ds
513/tcp	open	login
514/tcp	open	shell
1099/tcp	open	rmiregistry
1524/tcp	open	ingreslock
2049/tcp	open	nfs
2121/tcp	open	ccproxy-ftp
3306/tcp	open	mysql
5432/tcp	open	postgresql
5900/tcp	open	vnc
6000/tcp	open	X11
6667/tcp	open	irc
8009/tcp	open	ajp13
8180/tcp	open	unknown

MAC Address: 08:00:27:D0:B4:98 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds

(Ghostcat)

Effettuando una semplice scansione con **nmap** e possiamo vedere qual è la porta del processo ajp13

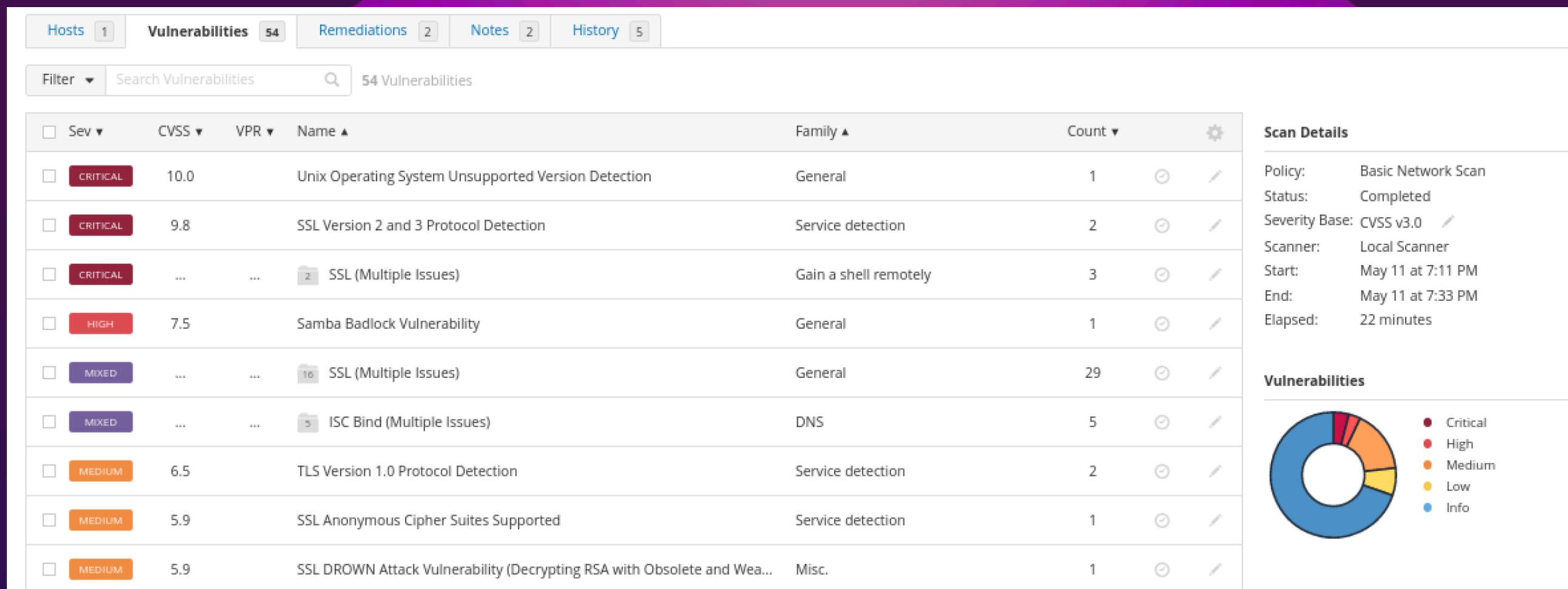
Con il comando
sudo netstat -tulpn | grep :8009
andiamo a vedere quale è il numero del
processo (PID) e lo andiamo a terminare
con il comando kill

```
msfadmin@metasploitable:~$ sudo netstat -tulpn | grep :8009
[sudo] password for msfadmin:
tcp        0      0 0.0.0.0:8009          0.0.0.0:*
4632/jsvc
msfadmin@metasploitable:~$ sudo kill 4632
msfadmin@metasploitable:~$ sudo netstat -tulpn | grep :8009
msfadmin@metasploitable:~$ _
```

LISTEN

Conclusion

Dopo aver effettuato i passaggi elencati precedentemente, per la risoluzione delle vulnerabilità, andiamo ad effettuare una nuova scansione e notiamo che le nostre azioni hanno avuto esito positivo e le criticità trovate sono notevolmente diminuite





REALIZZATO DA

MICHAEL ANDREOLI

OTMAN HMICH

STEFANO CESARONI